The need for rigorous software verification and validation of ShotSpotter

Jeanna Matthews (Clarkson University), Kevin Murray and Ronald Dean (TestPros) [1]
August 2022

1. Introduction

ShotSpotter is a sound detection system that claims to detect gunshots using a network of surveillance microphones and audio-recognition software using artificial intelligence. It is an example of software-based or algorithm-based technologies used increasingly throughout our society at all levels, including throughout the criminal justice system. As computer scientists and software engineers, we are intimately familiar with errors in software and the impact on society of those errors in critical software systems. The field of software engineering has developed a set of well understood best practices over decades to discover and repair errors in software. What is missing in the case of ShotSpotter is an insistence that these best practices be followed.

ShotSpotter has not been required to rigorously document essential testing data including exposing the method to important variables and documenting their impact, comparing developmental models to results in realistic deployed settings, and repetition when parts of the method or important variables change. They have not subjected their results to a reasonable level of independent repetition or independent review.

Once deployed, the methods ShotSpotter has used to report metrics like 97% accuracy, or 0.41% false positive rate do not provide a reasonable level of scientific rigor. Rather than

[1] The authors of this document are a professor of computer science who does research on probabilistic genotyping software where many of the same issues apply and professionals who regularly engage in software verification and validation of systems following the IEEE 1012 standard. The authors can be contacted at jnm@clarkson.edu, kmurray@testpros.com, and rdean@testpros.com.

rigorous testing, they are simply relying on error reports from deploying agencies. The Chicago Police Department's Report on the use of ShotSpotter technology, reports that for over 90% of the ShotSpotter alerts in 2020 and first half of 2021 (45,620 of the 50176 total alerts) police found no evidence of a gun crime including no victim, no witness, no blood, no gun, and no fired casings. However, if these were not reported to ShotSpotter then they are simply not reflected in the marketing literature provided by ShotSpotter. They have not reported false positive rate (FPR) and false negative rate (FNR) metrics under realistic conditions. Marketing literature and customer error reports are simply not a replacement for rigorous testing of critical software.

ShotSpotter has not been required to sufficiently test its software nor to document patterns of errors in the software prior to deployment. ShotSpotter continues to be deployed despite mounting evidence of unacceptable levels of inaccuracy[23]. This is not just our opinion. It represents the application of industry standards and widely accepted best practices such as IEEE 1012 "IEEE Standard for System, Software, and Hardware Verification and Validation". Industry standards, like IEEE 1012, are followed rigorously and routinely for many categories of critical software such as medical device software and aviation software. However, there has been a failure to apply these standards and best practices to ShotSpotter despite the fact it is undeniably a critical software system used to deploy armed officers and to produce evidence used in criminal proceedings.

[2] The Sun Sentinel reported that the Broward County (Florida) Sheriff abandoned their $500,000 Shotspotter system after discovering that the system was "wasting too much manpower sending deputies to false alarms" generated by firecrackers, car backfires, and other innocent noises. "Broward sheriff dropping gunshot detection system", I. Rodriguez, South Florida Sun-Sentinel, November 22, 2011.
[3]The Times Union reported that the Troy Police Department (Troy, New York) abandoned their Shotspotter system after determining that the system generated false alerts from innocent noises from a college campus and after determining that 911 calls were better at identifying actual gunfire. "Troy Will Turn Off Shotspotter", K. Crowe, Times Union, October 30, 2012.

2. Impact of Errors in Software Including Forensic Software

Most software flaws result from simple mistakes. The more complex the program, the greater the risk of flaws. Research has shown that professional programmers average six software defects for every 1000 lines of code (LOC) written[4]. In many situations, software faults are not discovered until they result in obvious and catastrophic failures. For example, NASA's Mars Climate Orbiter exploded because the software controlling its thrust was written to use English units instead of metric units[5]. Two Boeing 737 MAX aircraft crashed because a software modification made the aircraft vulnerable to nosedives[6]. An Ariane 5 rocket exploded because a fault in a program tried to "stuff a 64-bit number into a 16-bit space"[7]. A "software glitch" in Therac-25 radiation therapy machines resulted in cancer patients receiving excessive radiation when a certain group of commands was entered[8].

Recent history is littered with examples of latent flaws in forensic software and other software used in the criminal justice system being discovered after the software was used in numerous arrests and convictions. For example, notable flaws have been found in probabilistic genotyping (PG) software used to match DNA evidence found at crime scenes to possible suspects. In total, at least thirteen "coding faults" have been found in PG software, STRmix[9]. In one notable example, the miscode impacted 60 criminal cases, requiring new likelihood ratios to be issued in 24 cases[10]. Another PG program, FST, avoided independent review for years until a federal judge ordered source code disclosure[11]. The defendant's expert discovered that "[a] secret function . . . was present in the software, tending to overestimate the likelihood of guilt," and that "[t]he actual functioning of the software, revealed upon inspection of the source code, did not use the methodology publicly described in sworn testimony and peer-reviewed

[4] Hoang Pham, Software Reliability, in WILEY ENCYCLOPEDIA OF ELECTRICAL AND ELECTRONICS ENGINEERING 565, 565 (John G. Webster ed., 1999).
[5] Andrew Pollack, Missing What Didn't Add Up, NASA Subtracted an Orbiter, N.Y. Times (Oct. 1, 1999)
[6] Niraj Chokshi, House Report Condemns Boeing and F.A.A. in 737 Max Disasters, N.Y. Times (Sept. 16, 2020)
[7] James Gleick, Little Bug, Big Bang, N.Y. Times (Dec. 1, 1996).
[8] Fatal Radiation Dose in Therapy Attributed to Computer Mistake, N.Y. Times (June 21, 1986).
[9] STRmix, Summary of Miscodes https://bit.ly/36ILKWi (last updated Sept. 15, 2020).
[10] David Murray, Queensland Authorities Confirm 'Miscode' Affects DNA Evidence in Criminal Cases, Courier-Mail (Mar. 20, 2015).
[11] Stephanie J. Lacambra et al., Opening the Black Box: Defendants' Rights to Confront Forensic Software, The Champion, May 2018.

publications." Once these flaws were discovered, a high-profile conviction based on FST analysis was overturned[12] after FST developers had "aggressively resisted expert witness review that could have exposed the problem for 5 years while using the output of the system as evidence in over 1000 serious criminal cases."[13]

Similarly, in recent years, thousands of faults have been discovered in the source code of top breathalyzer systems. According to a 2019 New York Times investigation, breathalyzers "generate skewed results with alarming frequency," and "Judges in Massachusetts and New Jersey have thrown out more than 30,000 breath tests in the past 12 months alone."[14] After the New Jersey Supreme Court granted source code access, defense experts found that the code was "littered with "thousands of programming errors". Similar errors were found across the country.

There is no reason to think that ShotSpotter is somehow free of flaws that plague all software including forensic software. Without such a rigorous process, the evidence suggests that ShotSpotter does contain flaws and these flaws will simply go undiagnosed and unrepaired unless rigorous validation and verification is required and defendants are granted meaningful access to the source code.

3. Best Practices and Testing Guidance from IEEE

The field of software engineering has established clear standards and best practices that can be applied to find and repair many software flaws including documentation of requirements, specifications, test cases, test results, and maintenance records. There are many representations of the wide consensus in the software engineering community of best practices known to increase the reliability and accuracy of software systems including the Software

[12] Alan Feuer, Hasidic Man Convicted of Beating Black Student Gets Verdict Overturned, N.Y. Times (Oct. 10, 2018).
[13] See also Jeanna Neefe Matthews et al., When Trusted Black Boxes Don't Agree: Incentivizing Iterative Improvement and Accountability in Critical Software Systems, 2020 Proc. AAAI/ACM Conf. on AI, Ethics, & Soc'y 102, 103.
[14] Stacey Cowley & Jessica Silver-Greenberg, These Machines Can Put You in Jail Don't Trust Them, N.Y. Times (Nov. 3, 2019).

Engineering Institute (SEI)'s Capability Maturity Model (CMM)[15] and the Software Engineering Body of Knowledge (SWEBOK)[16]. In this section, we will highlight one concrete standard for verification and validation, IEEE 1012, that could be used to guide the level of review that could and should be required of ShotSpotter. IEEE 1012 is already widely accepted and applied to critical software throughout the US government including by the Department of Defense and the Food and Drug Administration. It would serve as a concrete standard to guide reasonable verification and validation of ShotSpotter.

The Institute of Electrical and Electronics Engineers, IEEE, is the largest organization of technology professionals in the world, representing more than 400,000 engineers, scientists, and allied professionals worldwide. IEEE-USA represents approximately 150,000 engineers, scientists, and allied professionals in the United States, many of whom are actively conducting research and development into artificial intelligence, software engineering, cybersecurity, and advanced computing, as well as other foundational and emerging technologies. IEEE is a leading developer of industry standards in a broad range of technologies. IEEE standards drive the functionality, capabilities, safety, and interoperability of products and services. IEEE 1012 Standard for System and Software Verification and Validation is one such standard.

3.1 Integrity Levels

IEEE 1012 prescribes different levels of verification and validation based on the risk and impact of deployed software. Specifically, IEEE 1012-2016 specifies a four-level schema summarized in Table 1[17]. The levels are defined by both the possible impact of an error (catastrophic, critical, marginal, or negligible) and the likelihood of error (reasonable, probable, occasional, or infrequent). Catastrophic impacts are defined as loss of human life, complete mission failure, loss of system security and safety, or extensive financial or social loss. ShotSpotter is used to deploy armed officers to the scene which could result in loss of human life and produces

[15] Capability Maturity Model (CMM) was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.
[16] The Software Engineering Body of Knowledge is an international standard ISO/IEC TR 19759:2005 specifying a guide to the generally accepted software engineering body of knowledge. The Guide to the Software Engineering Body of Knowledge has been created through cooperation among several professional bodies and members of industry and is published by the IEEE Computer Society.
[17] Other integrity schemas are acceptable. The International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/IEEE standard ISO/IEC/IEEE 15026-3:2015 describes the requirements and process for developing an integrity level schema.

evidence that could be used in court to guide decisions of incarceration which would cause extensive social loss.

The level of rigor applied to an IV&V activity, such as testing, is determined by the organization's Integrity Level schema. "Integrity level determination establishes the importance of the system to the user and acquirer based on complexity, criticality, risk, safety level, security level, desired performance, reliability, or other system unique characteristics." More rigorous IV&V is required when software, like ShotSpotter, is used to make critical decisions such as those in a criminal justice context.

Table 1: IEEE 1012-2016 Integrity Level Schema

Integrity Level	Description
4	Catastrophic consequences with reasonable, probable, or occasional or Critical consequences with reasonable or probable likelihood of occurrence
3	Catastrophic consequences with occasional or infrequent likelihood of occurrence or Critical consequences with probable or occasional likelihood of occurrence or Marginal consequences with reasonable or probable likelihood of occurrence
2	Critical consequences with infrequent likelihood of occurrence or Marginal consequences with probable or occasional likelihood of occurrence or Negligible consequences with reasonable or probable likelihood of occurrence
1	Critical consequences with infrequent likelihood of occurrence or Marginal consequences with occasional or infrequent likelihood of occurrence or Negligible consequences with probable, occasional, or infrequent likelihood of occurrence

3.2 Types of Testing and Documentation

IEEE 1012 specifies the level of verification and validation (V&V) testing dependent on the integrity level[18]. A V&V effort specifies and creates testing products (i.e., test plan, test design, test cases, and test procedures) and either conducts that testing or analyzes the results of that testing if it is conducted by another organization. For integrity level 4, the standard requires activities such as Software Component Testing, Software Integration Testing, Software Qualification Testing, and Software Acceptance Testing. The objectives of V&V testing are to identify errors. Acceptable V&V testing results, and absence of major errors, constitute the objective evidence for verifying and validating the system. This testing is designed to provide the objective evidence that the high integrity level requirements perform as intended under all operational scenarios including stress and abnormal conditions, error recovery situations, and high-risk hazardous situations.

The purpose of *System Integration* testing is to validate that each system element meets the system requirements and design, and that each element is successfully integrated with the other system elements and that requirements are traced to test design, test cases, and test results. A System Integration Test plan is developed based on system requirements and stakeholder requirements. The plan documents the test design, test cases and test results. IV&V evaluated the System Integration test plan in light of the requirements for:

> "i) Conformance to an increasingly larger set of functional requirements at each stage of integration.
> ii) Assessment of timing, sizing, and accuracy.
> iii) Performance at boundaries and under stress conditions.
> iv) Measures of requirements test coverage and system reliability."
> (IEEE 1012:2016, Table 1b 8.3 (5) 4)

IV&V also evaluates the test plan for conformance to the project's design document format and content. Lastly, IV&V evaluates the System Integration plan for traceability to system

[18] IEEE 1012-2016, Clause 6.2 V and V Testing.

requirements, external and internal consistency, test coverage, test standards and methods used, conformance to expected results, and feasibility of system integration, maintenance, and operation according to user needs. Design tasks include designing the tests based on system requirements and architecture models, continued tracing of the requirements from the test plan, verifying conformance to the project's defined document format and content, and validating the design against test plan criteria. Test cases are designed based on system requirements and design documents and continue tracing requirements from plan and design. IV&V verifies conformance to the project's defined document format and content and validates the test cases against test plan and design criteria. Test procedures are developed from the test cases, system requirements and design documents. The test procedures continue tracing the requirements and verifying conformance to the project's defined document format and content. System Integration testing is executed against the hardware elements, software elements, and other external elements to validate that the system satisfies the test criteria. The test is conducted based on the sequence of test procedures, and the strategy of test scenario inputs. Test results as well as discrepancies between actual and expected results are documented according to the test plan.

The purpose of *System Qualification* testing is to validate the integrated system against the system requirements. The System Qualification testing follows the same pattern as System Integration testing with the focus on satisfaction of functional, performance, security, operation, and maintenance requirements in the system's environment as described in the system's concept documents. During System Qualification testing, IV&V also evaluates the system documentation, such as training materials, procedures and changes to procedures, and the user guide. IV&V also evaluates performance, stress testing, code coverage, and results during System Qualification testing. Lastly, IV&V evaluates the test documentation for conformance to the project's defined document format and content.

The purpose of *System Acceptance* testing is to validate that the system correctly implements the requirements in the intended operating environment or as near to the operating environment as possible. The Systems Acceptance testing follows the pattern of System Integration and System Qualification testing with the focus on test tasks that demonstrates conformance to the requirements in the operational environment and test coverage of system requirements. Test tasks are based on system requirements, concept documentation, and architecture models and bring everything together to validate that the system performs as required when it is installed in

the operational environment and staffed by operators. Subjective tests based on attitudes and experience are conducted to demonstrate customer satisfaction. As with the other tests, the results as well as discrepancies between actual and expected results are documented according to the test plan.

Concept documentation includes: System requirements, Software requirements specification(s), Software interface requirements, specification(s), Hardware requirements specification(s), System design, Software design documents(s), Hardware drawings, System element(s), Software traceability analysis, and Hardware traceability analysis. IV&V evaluates concept documentation to assure requirements are complete and reflect stakeholder needs. In many systems these documents contain hidden requirements which if left undocumented, become crucial to the overall validation of the system.

Database analysis and dataflow analysis are optional activities performed during Stakeholders needs and requirements. For a Level 4 system, these tasks are rarely optional and would be traced through any evaluation and test of the Stakeholders requirements.

Source code analysis or inspection is performed under Clause 9.4 of IEEE 1012. The purpose of the inspection is to validate that the code and its documentation is correct, consistent, complete, accurate, readable, and testable. The inspected source code becomes part of the system requirements and is traced through any evaluation of system requirements.

3.3 Independence of Testing

The software engineering community recognizes that the V&V process must be independent to avoid conflicts of interest. Specifically, letting developers certify their own software is a clear conflict of interest and leads to well-recognized problems. IEEE 1012 specifies three important categories of independence for independent verification and validation (IV&V)- technical, managerial, and financial.

Specifically, *technical independence* “[r]equires the IV&V effort to use personnel who are not involved in the development of the system or its elements. The IV&V effort should formulate its own understanding of the problem and how the proposed system is solving the problem.”

“Technical independence means that the IV&V effort uses or develops its own set of test and analysis tools separate from the developer’s tools.” And if sharing tools is necessary, “IV&V conducts qualification tests on tools to assure that the common tools do not contain errors that may mask errors in the system being analyzed and tested.” This independence requires the exclusion of parties with a stake in the outcome, which for forensic technologies includes forensic labs and police departments that, while not financially dependent on developers, have a shared interest in software’s acceptance.

Managerial independence “[r]equires that the responsibility for the IV&V effort be vested in an organization separate from the development and program management organizations. Managerial independence also means that the IV&V effort independently selects the segments of the software, hardware, and system to analyze and test, chooses the IV&V techniques, defines the schedule of IV&V activities, and selects the specific technical issues and problems to act on.” The IV&V effort must be “allowed to submit to program management the IV&V results, anomalies, and findings without any restrictions (e.g., without requiring prior approval from the development group) or adverse pressures, direct or indirect, from the development group.”

Financial independence “[r]equires that control of the IV&V budget be vested in an organization independent of the development organization. This independence prevents situations where the IV&V effort cannot complete its analysis or test or deliver timely results because funds have been diverted or adverse financial pressures or influences have been exerted.”

Further, there are five forms of managing independence: Classical, Modified, Integrated, Internal, and Embedded.

Classical IV&V encompasses all the independence parameters, technical, managerial, and financial, and is usually performed by a separate organization that works closely with development, but reports to a separate manager and has a separate budget. Thus IV&V results are rendered without outside influence. “Classical IV&V is generally required for integrity level 4 (i.e., loss of life, loss of mission, significant social loss, or financial loss) through regulations and standards imposed on the system development.” (IEEE 1012-2016, Annex C, C.2.2)

Modified IV&V is used when the organization selects a company (often called the prime integrator) to manage the overall progress of the project. This prime integrator then selects other organizations to assist with all aspects of the project including IV&V. Since IV&V reports to the prime integrator, managerial independence is compromised. However, technical independence and financial independence are preserved because IV&V results are still without bias or financial pressure. "Modified IV&V effort would be appropriate for systems with integrity level 3 (i.e., an important mission and purpose)." (IEEE 1012-2016, Annex C, C.2.3)

Integrated IV&V is focused on providing rapid IV&V results to the development process. IV&V works side by side with development in reviewing interim work products, and provides feedback during inspections, walkthroughs, and reviews. Managerial and financial independence is maintained although technical independence is compromised. "Impacts to technical independence are counterbalanced by benefits associated with a focus on interdependence between the integrated IV&V effort and the development organization. Interdependence means that the successes of the organizations are closely coupled, ensuring that they work together in a cooperative fashion." (IEEE 1012-2016, Annex C, C.2.4)

Internal IV&V uses personnel from within the development organization ideally not associated with the development effort. All the IV&V parameters are compromised due to managerial reporting, peer pressure to lessen the detail of analysis, or reduced funding limits due to development pressure to complete the project. Internal IV&V does have the benefit of readily available staff who know the system and the software. "This form of IV&V is used when the degree of independence is not explicitly stated, and the benefits of preexisting staff knowledge outweigh the benefits of objectivity." (IEEE 1012-2016, Annex C, C.2.5)

Embedded IV&V is used to assure conformance to processes and procedures. It is similar to Internal IV&V in that personnel from the development organization who are not directly involved with the development effort perform the IV&V activities. Technical independence is compromised because IV&V is not asked to analyze the solution. Managerial and financial independence is compromised as in Internal IV&V. The benefits of Internal IV&V apply to Embedded IV&V.

4. Shot Spotter Assessment

ShotSpotter is a sound detection system that claims to detect gunshots using a network of surveillance microphones and audio-recognition software using artificial intelligence. When microphones deployed by ShotSpotter detect an impulsive noise event, data collected about the noise event, including sound waves and the recorded audio of the event, are sent to a computer algorithm and to a customer service agent. ShotSpotter's algorithm has two important tasks when analyzing this data: determining whether the noise originated from a gunshot or some other source and offering a location estimate for the noise event. The results of the ShotSpotter algorithm are conveyed to human employees who are authorized to override the determinations of the algorithm and substitute their own judgments. Both the algorithmic determinations and the human override process are designed to occur within 60 seconds of the noise event. ShotSpotter then rapidly notifies local police departments so that officers can respond quickly to the scene of a noise event.

ShotSpotter released a document "Independent Audit of the ShotSpotter Accuracy" prepared by Edgeworth Analytics on July 22, 2021, that claims 97.59% overall accuracy rate and a 0.41% false positive rate. However, we have serious concerns about their methodology. Specifically, these figures do not reflect controlled testing. They simply reflect error reports received by ShotSpotter. It does not compare ShotSpotter reports to controlled known test cases (e.g., a gun fired at a precise known location in a realistic test setting, a non-gunshot noise generated at a precise known location[19]). More specifically, it would be impossible to report accuracy with this methodology because there would be very little visibility into gunshot events that are missed by ShotSpotter. Furthermore, there is no way to know how many false positives go unreported by police. The Chicago Police Department's Report on the use of ShotSpotter technology, reports that for over 90% of the ShotSpotter alerts in 2020 and first half of 2021 (45,620 of the 50176 total alerts) police found no evidence of a gun crime including no victim, no witness, no blood, no gun, and no fired casings. This suggests that most false positives simply go unreported and would therefore not be reflected by the methodology used by Edgeworth.

[19] In November 2021, Singh et al published the peer reviewed article "Data Collection, Modeling, and Classification for Gunshot and Gunshot-like Audio Events: A Case Study" in which they assembled a data set of plastic bag pop sounds that could be used to test systems like ShotSpotter for false positives.

Further, in April 2021, a subpoena was issued to ShotSpotter requesting ". . .all Deployment Qualification testing, live-fire testing, or similar internal validation testing of the ShotSpotter gunshot detection system deployed in Chicago. . .". In response, Mike Will, Vice President of ShotSpotter, stated that ""No live fired or DQV testing was performed in any district as part of this service. No documentation or materials exist."

ShotSpotter has not been required to provide evidence of rigorous testing in realistic urban environments where it is deployed. This is quite different from testing under highly controlled conditions. Realistic testing environments would include variable levels of environmental noise such as traffic and construction noises as well as reflection, refraction, and diffraction of sound waves from a complex set of obstacles such as buildings, fences, and other structures. Common sense alone leads us to doubt that the challenging problem of acoustical classification and location in a complex urban environment would have a false positive rate of only 0.41% and we have seen no scientifically credible evidence to support the claim.

5. Conclusion

It is imperative for public safety that criminal justice software like ShotSpotter be held to reasonable standards of testing, verification, and validation before deployment. While that has not happened, there is still an opportunity to do so now. IEEE 1012, a standard already followed widely throughout the US Government to validate critical software, provides a concrete map for independent verification and validation. An independent source code review would be an important first step towards holding ShotSpotter to the industry standards of verification and validation applicable to critical software.